Pusan National University
Computer Science and Engineering
Technical Report 2022-0X

블록체인 기반의 신재생에너지 거래 플랫폼 개발

201724595 차민준
201724436 김재현
201724508 오재석

지도교수
최 윤 호

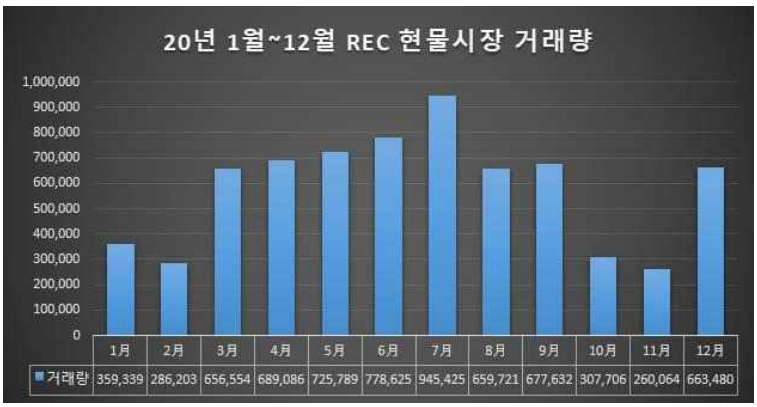
목차

1. 서론

1.1 연구 배경

[그림 1] re100

- 기후변화가 우리 삶에 점점 영향을 끼치면서 피해가 커지고 있다. 그에 대한 대응책으로 기업에서 사용하는 전력의 100%를 재생에너지로 대체하자는 국제적 기업간 협약프로젝트인 RE100이 등장하였다. RE100의 이행수단으로 우리나라와 북미는 REC코인을 발행하여 REC거래시장이 형성되고 REC발급과 유통에 대한 안전한 플랫폼의 필요성이 생겼다.



[그림 2] 2020년 REC거래량 추이

< 연도별 의무비율 입법예고 안 (시행령 별표3) >

연 도	'22년	'23년	'24년	'25년	'26년 이후
의무비율	12.5%	14.5%	17.0%	20.5%	25.0%

[그림 3] 신재생에너지공급 비율

- 2020년, 2021년에도 REC코인의 거래는 활발하게 일어났고 2022년부터는 국내 대규모 발전소에 적용되는 신재생에너지공급 의무화(RPS) 비율도 늘어나면서 REC시장은 더욱 확대될 전망이다.

- 블록체인은 일종의 데이터 보안 기술로 아주 강력한 데이터 보안 솔루션이다. 조직의 승인 없이 누구든지 블록체인 네트워크에 참여가 가능한 개방형 블록체인(publici blockchain)과 법적 책임을 지는 허가받은 사람만 블록체인 네트워크에 참여할 수 있는 블록체인인 프라이빗 블록체인(private blockchain) 있다. 최근 프라이빗 블록체인은 기업체 단위에서 사용하기에 장점이 많아 각광받고 있다.

- 본 졸업과제에서는 REC거래를 블록체인을 사용하여 안전하게 거래할 수 있는 플랫폼을 만드는 것을 목적으로 한다.

1.2 기존 문제점

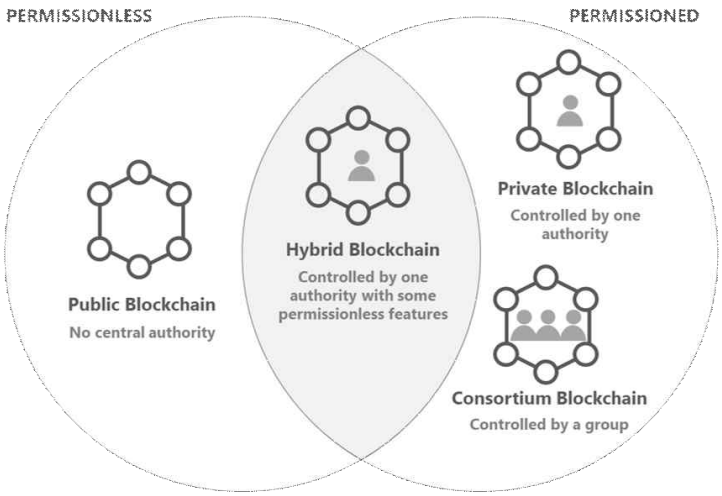
[그림 4] 현재 REC거래 과정

- 현재의 REC거래는 비효율적인 모습을 보여준다. 사업자가 REC를 등록하면 전력거래소에서 매물로 설정하고 공급 의무자는 REC를 돈을 지불하고 구매하고 대금청구 및 대금이 오고 간다. 국세청에 대금 수수료가 납부되고 이 모든 과정이 완료되면 REC의 소유권이 이전된다. 이때 각 기관들이 소유하고 있는 데이터들이 서로 달라 거래 시에 서로 대조가 필요하며 거래 과정도 복잡해진다. 입찰 참여부터 최종 수수료 납부까지 4개 기관, 10단계를 거치는 과정으로 되어 있으며, REC구매 절차가 매우 복잡해 이용자의 컴플레인이 많았다. 에너지 제조산업 특성상 고령층 및 농민 사업자들의 비율이 높아 그 문제점은 더욱 심각하다.

각 기관의 데이터베이스에 위조, 해킹 등의 위협도 존재한다. 직접적으로는 거래 내용의 해킹으로 금전적인 피해가 발생할 수 있고, 간접적으로는 이런 공격이 감지되었을 경우에 REC거래 자체가 일시중단되면서 피해가 발생할 수 있다. 또 기관들이 각자 소유하고 있는 데이터베이스에 대해 발급내역과 대금정보가 불일치할 가능성이 있어 거래 소모비용이 증가한다. 현재까지는 이런 절차적 문제들로 인해 지정된 장소에 직접 방문해야 할 경우가 있었고 서면계약 체결이 필요했다.

1.3 연구 목표

1.3.1 하이퍼래저



[그림 5] 블록체인의 종류

- 프라이빗 블록체인은 법적 책임을 지는 허가받은 사람만 블록체인 네트워크에 참여할 수 있는 블록체인이다. 운영과 참여의 주체가 분명하기 때문에 코인을 발행하지 않아도 되고 속도가 빠르다. 합의 과정에 참여할 수 있는 참여자가 미리 지정되어있고 필요에 따라 추가하거나 제거할 수 있어 사업에 적합한 블록체인이라는 평가를 받는다.

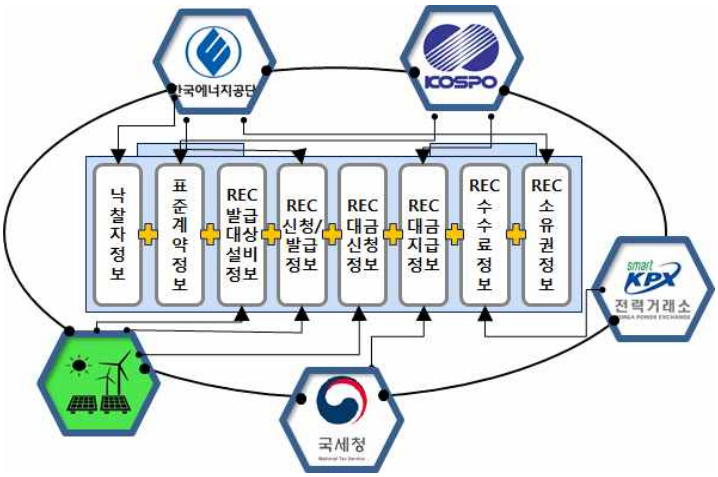
  블록체인 기술을 사용하지만 참여에 대한 권한은 중앙기관에서 가지고 있으므로 중앙이 힘을 가지고 있고 분산된 데이터베이스만 갖는 구조로 블록체인의 보안 검증성의 특징만 사업으로 가져온 모양새를 띈다. 본 과제에서는 REC의 거래를 위한 플랫폼을 만드는 것을 목적으로 하는데, REC거래에서는 신뢰받는 기관만이 참여를 해야하므로 프라이빗 블록체인이 적합하다.

[그림 6] 하이퍼래저 패브릭

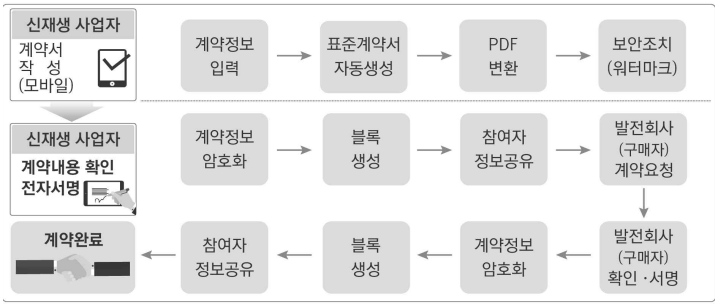
- 하이퍼래저는 리눅스 기반의 오픈소스 프로젝트로 허가받은 사람만이 접근할 수 있는 프라이빗 블록체인(private blockchain)이다. 비트코인이나 이더리움처럼 암호화폐 중심이 아닌 블록체인의 보안성이 필요한 물류, 제조, 기술, 금융산업등 도입 가능한 블록체인 기술 표준을 제시하는 것에 목표를 두고 있다. iroha, burrow, grid 등이 있지만 그중에서 이더리움에서 좋은 평가를 받았던 chain-code를 활용해서 작업에 대한 규칙이 있는 smart-contract 방식을 사용하는 하이퍼래저 패브릭(hyperledger-fabric)을 사용하도록 한다

  체인코드를 사용하여 블록체인 네트워크의 각 구성원(Organizations)들과 사용자(Users)에 따라 체인코드에 접근하는 권한을 설정해줄 수 있고 네트워크 조직들의 노드에 접근하여 거래가 가능해진다.

1.3.2 연구 목표



[그림 7] 정부가 추진 중인 블록체인 기반 REC거래 플랫폼



[그림 8] 정부가 추진 중인 블록체인 기반 REC거래 과정

- 본 과제는 프라이빗 블록체인 기술인 하이퍼래저 패브릭(hyperledger-fabric)을 사용하여 안전한 Rec코인 거래 플랫폼을 구축하는 것을 목표로 한다.
  허가된 사용자에 한정하여 네트워크에 접근할 수 있도록 하고 각 인증기관을 네트워크에 참여시켜 서로 분리되어있는 데이터베이스가 아닌 블록체인 네트워크에서 데이터를 직접 받아올 수 있도록 한다.
  하나의 블록체인 네트워크 안에서 구매자, 판매자, 전력거래소, 국세청 조직을 만들고 체인코드에 따라서 통신이 가능하도록 하고, 웹페이지를 기반으로 거래할 수 있도록 한다.

2. 연구 배경

2.1 연구 개발 환경

2.1.1 개발 언어

- **Go**
  하이퍼래저 패브릭에서는 smart-contract의 방법으로 chaincode를 사용한다.
  체인코드를 사용해서 블록체인 네트워크 조직의 계약을 진행할 수 있다.

- **shellscript**
  블록체인 config파일을 기반으로 docker를 실행시키고 체인코드의 패키징와 설치, 커밋을 위해서는 하이퍼래저 패브릭에서 배포하는 binary파일을 실행시켜야 한다. 네트워크의 간편
  한 실행을 위해서 사용하였다.

- **java**
  Back-end
  블록체인 서버와 통신하는 Rest api제작을 위해 사용하였다.

- **웹 (javascript,css,html)**
  Front-end를 만들기 위해 사용하였다.

2.1.2 프레임워크

- **Spring boot**
  블록체인 서버와 통신하는 Rest api를 위해서 사용하였다
- **Reactjs**
  프론트를 제작할 때 사용하였다.
- **Docker**
  하이퍼래저 조직/체인코드/CA를 실행시킬 때 사용하였다.
- **MySql, CouchDB**
  각각 유저 정보 저장, 블록체인 레저 저장할 때 사용하였다.

2.1.3 블록체인 네트워크 실행 환경

- macOS : vsrsion 12.6
- Docker : 20.10.17
- Docker-compose : v2.10.2
- Go : go1.19 darwin/arm64

3. 연구 내용

3.1 구상

본 과제는 REC를 안전하게 거래할 수 있는 플랫폼을 만드는 것을 목표로 한다.
현재 REC거래에 참여하는 조직과 요구사항은 다음과 같다.

1. 국세청
   국세청은 발전사업자와 공급의무자 사이에 계약이 체결되고 정산이 진행될 때 세금을 징수한다. 세율을 정할 수 있어야 한다.

2. 전력거래소
   전력거래소는 모든 거래가 열람 가능해야 한다.
   발전사업자와 공급의무자 사이에 모든 거래의 열람이 가능해야 하지만 수정은 할 수 없다.

3. 발전사업자(판매자)
   전력을 생산해서 REC를 발행하고 판매할 수 있어야 한다.
   판매 개수와 REC의 개당 가격을 설정할 수 있어야 한다.
   본인의 판매 기록은 볼 수 있지만 다른 사람들의 거래 기록은 열람할 수 없다.
   판매한 금액에 대한 환전이 가능해야 한다.

4. 공급의무자(구매자)
   REC를 구매할 수 있어야 한다.
   본인의 구매 기록은 볼 수 있지만 다른사람들이 거래 기록은 열람할 수 없다.
   돈을 충전할 수 있어야 한다.

각 역할별로 저장해야할 정보는 다음과 같다.

조직	정보	
발전사업자	ID : string,	(식별자 정보)
	REC : int,	(보유 REC개수)
공급의무자	KRW : int,	(보유액)
	role : string,	(역할 : seller/buyer)
전력거래소	ID : string,	(식별자 정보)
	REC : int,	(가격)
	KRW : int,	(금액)
	seller : string,	(판매자)
	buyer : string,	(구매자)
	state : string,	(상태)
	time : string	(시간)
국세청	tax : int	(세금)

[표 1] 기관별 저장 데이터

발전사업자와 공급의무자는 사람의 정보를 저장한다.
사람의 식별자인 ID와 가지고 있는 보유액과 REC의 개수, 역할을 알 수 있어야 한다.

전력거래소는 매물의 정보가 저장되는데, state에 따라 두 가지로 나뉜다.

- SALE 현재 판매 중인 매물
  ID와 seller에는 같은 판매자 정보가 기록된다.
  buyer에는 정보가 들어가 있지 않다.
  KRW와 REC는 판매하고 있는 REC개수 / REC 한 당 가격이 기록된다.
  time에는 가장 최근 판매자가 판매하는 rec의 개수를 늘리거나 가격을 변화시켰을 때로 갱신된다.

- DONE : 판매 완료된 매물
  ID에는 무작위로 생성된 해시(거래번호)가 기록된다.
  REC에는 거래량이 기록된다.
  KRW는 거래금액이 기록된다.
  seller에는 판매자 정보가 들어간다.
  buyer에는 구매자 정보가 들어간다.
  time에는 거래시간이 기록된다.

국세청은 현재의 세금정보가 들어간다.
거래 시 거래금액에 대한 세금을 징수한다.

3.2 조직

조직(organization)은 블록체인 네트워크에 참여하는 하나의 사용자그룹 단위이다.
조직별로 노드(peer)를 운영하고, 하나의 조직에는 여러 명의 사용자를 가질 수 있다.
사용자는 개인의 msp(인증파일)를 통해서 조직의 일원임을 인증받고 체인코드에 접근할 수 있다.

본 네트워크에서는 필요한 각 기관을 하나의 조직으로 설정하였다.

Organizations			
buyer	seller	koreapower	tax
구매자 기관	판매자 기관	한국전력 기관	국세청 기관

3.3 채널과 체인코드

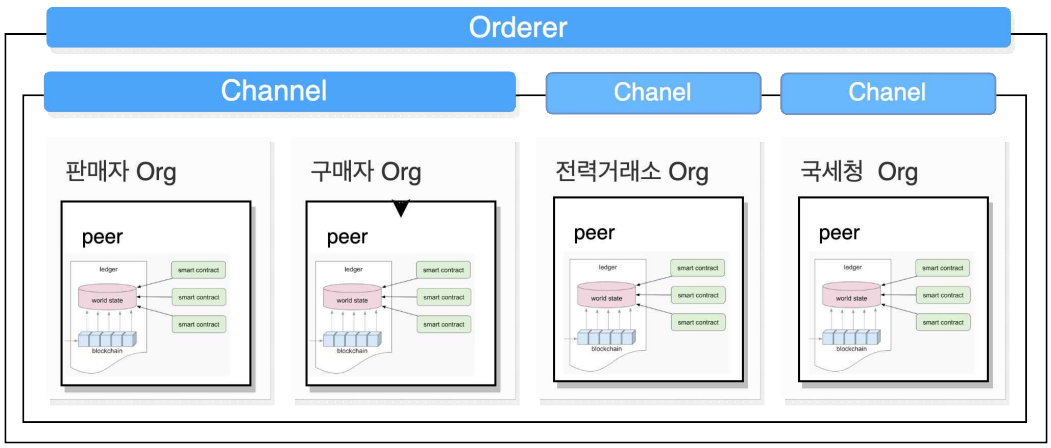
채널은 하이퍼래저 패브릭에서 사용하는 그룹이다.
한 채널당 한 개의 장부를 가지고 있고 해당 채널에 가입한 조직들은 한 장부의 복사본을 가진다. 이 장부를 수정하려면 여러 피어들의 합의(consensus)를 요구하고 조직의 접근권한을 설정해줄 수 있다. 채널에 가입된 조직(organization)만이 해당 채널에 설치된 체인코드를 통해 데이터에 접근하고 csmart-contract가 가능해진다.
따라서 비슷한 역할의 조직들을 묶어 체인코드를 설치하여 기능을 부여할 수 있고 해당 채널에 가입하지 않은 다른 기능을 가진 조직은 해당 데이터에 접근할 수 없다.

위 [표 1]를 바탕으로 비슷한 조직끼리 묶으면 다음과 같다.

buyer	seller	koreapower	tax
구매자 기관	판매자 기관	한국전력 기관	국세청 기관
people		market	revenue

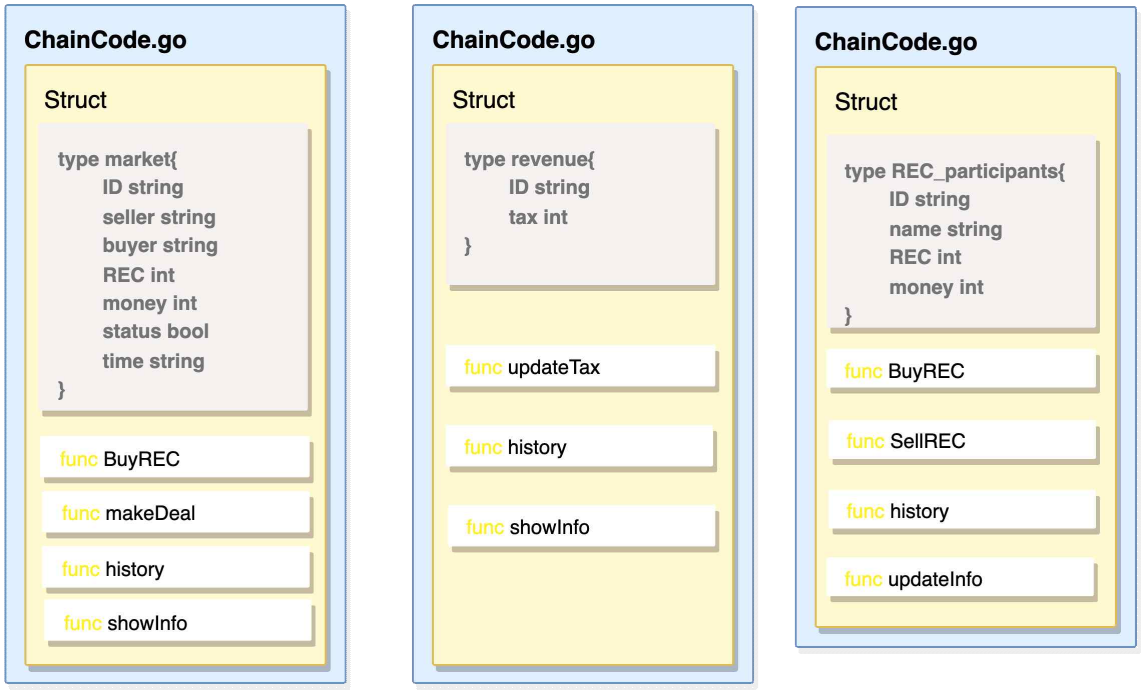
위 표를 바탕으로 3가지 채널을 구성하였다.

Channel people : 사람에 대한 정보를 관리
Channel market : 전력거래소에 대한 정보를 관리
Channel revenue : 세금에 대한 정보를 관리
각 채널에는 한 개의 체인코드를 설치하는 것으로 한다.



[그림 9] 네트워크 구성

구성된 네트워크는 [그림 9]와 같다.



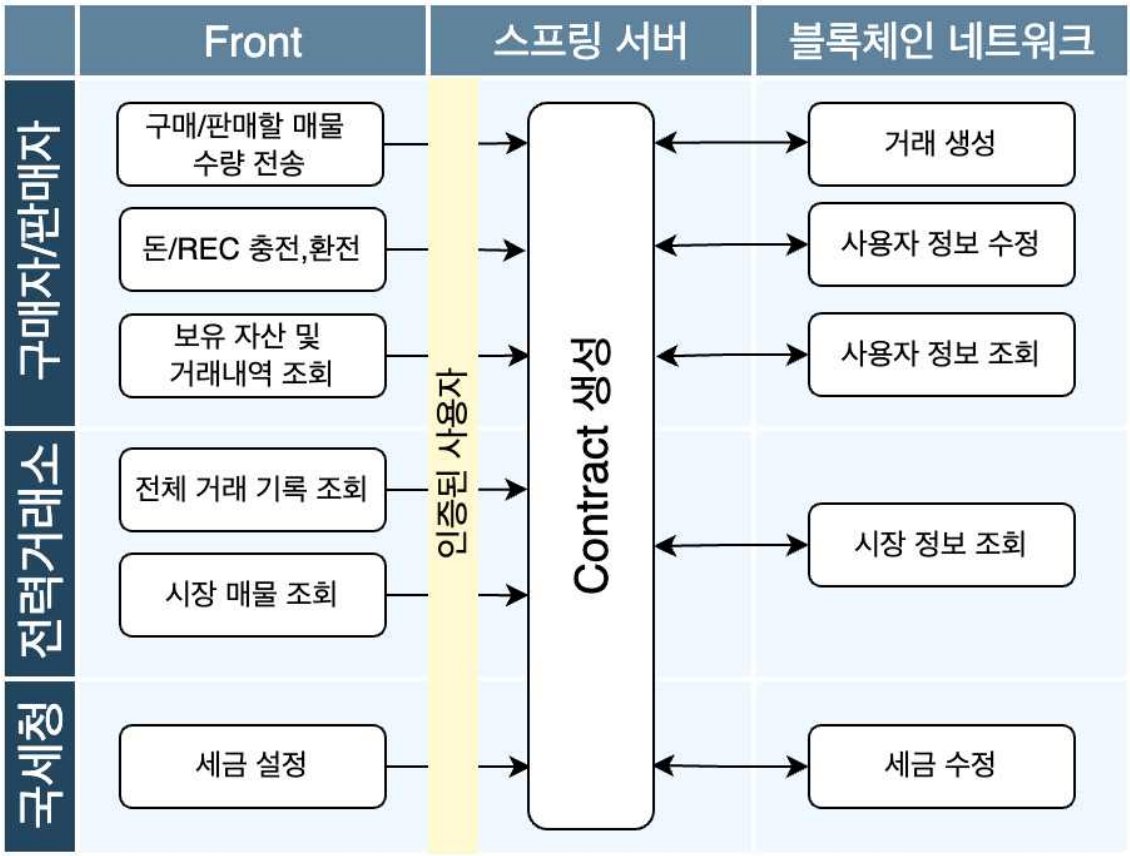
[그림 10] 각 채널의 체인 코드

각 채널의 체인코드는 [그림 10]과 같다.

정리하면 본 과제의 블록체인은 다음과 같다.

블록체인 네트워크				
오더러	1개의 오더러			
조직	buyer	seller	koreapower	tax
피어(앵커 피어1개)	buyer.peer0.example.com	seller.peer0.example.com	koreapower.peer0.example.com	tax.peer0.example.com
사용자	admin User1 User2 User3	admin User1 User2 User3	admin	admin
채널	people		market	revenue
체인코드	people		market	revenue

3.4 전체 시나리오



[그림 11] 전체 시나리오

구상한 조직, 체인코드, 채널에 따라 사용자별로 시나리오를 정리하면 [그림 11]과 같다.

Contract에는 체인코드를 실행하기 위한 사용자의 보안 키가 들어간다.

4. 연구 결과 분석 및 평가

제한점

- 본 과제에서는 사용자 명수를 지정해 놓는다. CA를 생략하고 하이퍼래저 패브릭에서 제공하는 기본 암호화 파일인 crytogen을 이용하여 암호화 파일을 생성하는 것으로 한다.
  따라서 회원가입이 제한되고, 조직들의 사용자는 네트워크 설정을 따라간다.
- 본 과제에서는 스프링 서버에 암호화 파일을 직접 보내주지 않는다. 로컬 환경에서 모든 정보가 있는 것을 가정하여 실행하였다.
- 본 과제에서는 조직의 구성을 피어 하나로 제한하였다. 본래는 여러개의 피어와 앵커피어(anchor peer)로 구성할 수 있으나, 본 과제에서는 편의를 위해 앵커피어 하나로 실행하였다.

4.1 블록체인 네트워크

4.1.1 설정

4.1.1.1 제네시스 블록 설정

```
Organizations:

    # SampleOrg defines an MSP using the sampleconfig.  It should never be used
    # in production but may be used as a template for other definitions
    - &OrdererOrg
        # DefaultOrg defines the organization which is used in the sampleconfig
        # of the fabric.git development environment
        Name: OrdererOrg

        # ID to load the MSP definition as
        ID: OrdererMSP

        # MSPDir is the filesystem path which contains the MSP configuration
        MSPDir: ../organizations/ordererOrganizations/example.com/msp

        # Policies defines the set of policies at this level of the config tree
        # For organization policies, their canonical path is usually
        #   /Channel/<Application|Orderer>/<OrgName>/<PolicyName>
        Policies:
            Readers:
                Type: Signature
                Rule: "OR('OrdererMSP.member')"
            Writers:
                Type: Signature
                Rule: "OR('OrdererMSP.member')"
            Admins:
                Type: Signature
                Rule: "OR('OrdererMSP.admin')"

        OrdererEndpoints:
            - orderer.example.com:7050
```

[그림 13] configtx.yaml의 일부

- configtx/configtx.yaml파일을 수정하여 네트워크의 상세 내용을 설정해줄 수 있다.

해당 파일을 사용해서 블록체인 네트워크의 gennesis.block이 생성된다.

[그림 13]와 같이 오더러, buyer(조직), seller(조직), tax(조직), market(조직)을 정의해준다.

```
Profiles:

    people:
        <<: *ChannelDefaults
        Orderer:
            <<: *OrdererDefaults
            Organizations:
                - *OrdererOrg
            Capabilities: *OrdererCapabilities
        Application:
            <<: *ApplicationDefaults
            Organizations:
                - *seller
                - *buyer
            Capabilities: *ApplicationCapabilities

    revenue:
        <<: *ChannelDefaults
        Orderer:
            <<: *OrdererDefaults
            Organizations:
                - *OrdererOrg
            Capabilities: *OrdererCapabilities
        Application:
            <<: *ApplicationDefaults
            Organizations:
                - *tax
            Capabilities: *ApplicationCapabilities

    market:
        <<: *ChannelDefaults
        Orderer:
            <<: *OrdererDefaults
            Organizations:
                - *OrdererOrg
            Capabilities: *OrdererCapabilities
        Application:
            <<: *ApplicationDefaults
            Organizations:
                - *koreapower
            Capabilities: *ApplicationCapabilities
```

[그림 14] configtx.yaml Profiles

[그림 14]은 configtx.yaml의 Profiles로, 네트워크를 실행할 때 처음 만들어지는 genesis block을 설정해준다.

[그림 14]와 같이 각각의 조직을 정의해준 후 프로파일을 사용해 제네시스 블록의 구조를 정의한다. people에는 조직으로 seller와 buyer 모두가 들어가 있는 것을 확인할 수 있다.

[그림 15] 제네시스 블록

이후 네트워크 실행 시 channel-artifacts 파일에서 생성된 제네시스 블록을 확인할

수 있다.

4.1.1.2 도커 설정

- 하이퍼래저 패브릭은 도커 이미지를 사용해여 도커 컴포즈로 연결되어있는 도커 컨테이너를 올린다. 이 컨테이너들은 서로 포트를 통해 연결되어있어야 하므로 도커 컴포즈파일 설정과 도커 포트 설정이 필요하다.

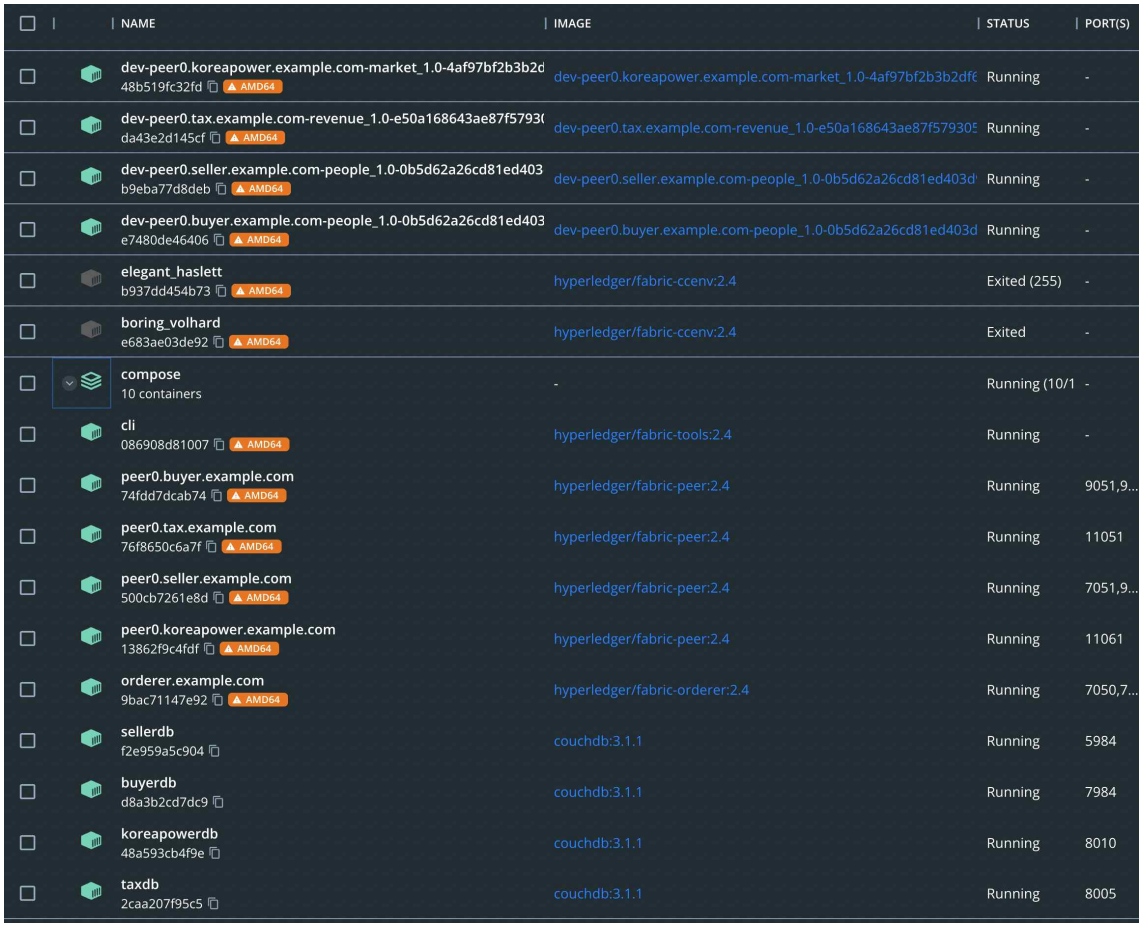
[그림 16] Docker images

- 하이퍼래저 패브릭은 Docker-compose를 이용하여 각 조직, CA, 데이터베이스를 도커로 실행시키고 연결시켜 실행된다.
  도커는 4개의 도커 이미지 파일을 사용한다.

  Hyperledger/fabric-peer : 하이퍼래저 피어 이미지
  Hyperledger/fabric-tools : 하이퍼래저 툴
  Hyperledger/fabric-orderer : 하퍼래저 오더러
  couchdb : 카우치디비

[그림 17] 도커 설정 파일

- 도커 이미지들에 대한 설정은 compose-test-net.yaml에서 설정이 가능하다.
  생성할 이미지/이름/환경변수/포트 설정이 이루어진다.



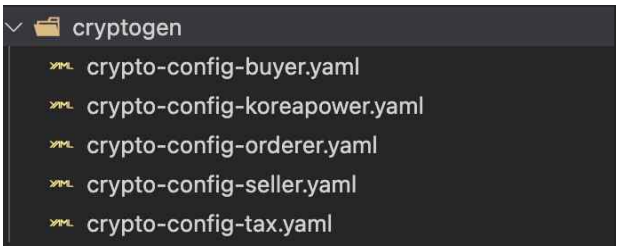
	NAME	IMAGE	STATUS	PORT(S)
☐	dev-peer0.koreapower.example.com-market_1.0-4af97bf2b3b2d 48b519fc32fd AMD64	dev-peer0.koreapower.example.com-market_1.0-4af97bf2b3b2df	Running	-
☐	dev-peer0.tax.example.com-revenue_1.0-e50a168643ae87f5793(da43e2d145cf AMD64	dev-peer0.tax.example.com-revenue_1.0-e50a168643ae87f57930	Running	-
☐	dev-peer0.seller.example.com-people_1.0-0b5d62a26cd81ed403 b9eba77d8deb AMD64	dev-peer0.seller.example.com-people_1.0-0b5d62a26cd81ed403d	Running	-
☐	dev-peer0.buyer.example.com-people_1.0-0b5d62a26cd81ed403 e7480de46406 AMD64	dev-peer0.buyer.example.com-people_1.0-0b5d62a26cd81ed403d	Running	-
☐	elegant_haslett b937dd454b73 AMD64	hyperledger/fabric-ccenv:2.4	Exited (255)	-
☐	boring_volhard e683ae03de92 AMD64	hyperledger/fabric-ccenv:2.4	Exited	-
☐	compose 10 containers	-	Running (10/1	-
☐	cli 086908d81007 AMD64	hyperledger/fabric-tools:2.4	Running	-
☐	peer0.buyer.example.com 74fdd7dcab74 AMD64	hyperledger/fabric-peer:2.4	Running	9051,9...
☐	peer0.tax.example.com 76f8650c6a7f AMD64	hyperledger/fabric-peer:2.4	Running	11051
☐	peer0.seller.example.com 500cb7261e8d AMD64	hyperledger/fabric-peer:2.4	Running	7051,9...
☐	peer0.koreapower.example.com 13862f9c4fdf AMD64	hyperledger/fabric-peer:2.4	Running	11061
☐	orderer.example.com 9bac71147e92 AMD64	hyperledger/fabric-orderer:2.4	Running	7050,7...
☐	sellerdb f2e959a5c904	couchdb:3.1.1	Running	5984
☐	buyerdb d8a3b2cd7dc9	couchdb:3.1.1	Running	7984
☐	koreapowerdb 48a593cb4f9e	couchdb:3.1.1	Running	8010
☐	taxdb 2caa207f95c5	couchdb:3.1.1	Running	8005

[그림 18] 블록체인 네트워크 도커 컴포즈

실행 결과물이다. compose안에 조직의 컨테이너들이 실행된다.

4.1.1.3 암호파일 설정

- 각 조직과 조직의 사용자들이 사용할 암호화 파일이 필요하다.
  본 과제에서는 하이퍼래저 패브릭이 기본 제공하는 cryptogen파일을 사용한다.
  cryptogen.yaml 파일을 사용해서 User의 숫자를 설정하고 암호화파일을 생성할 수 있다.

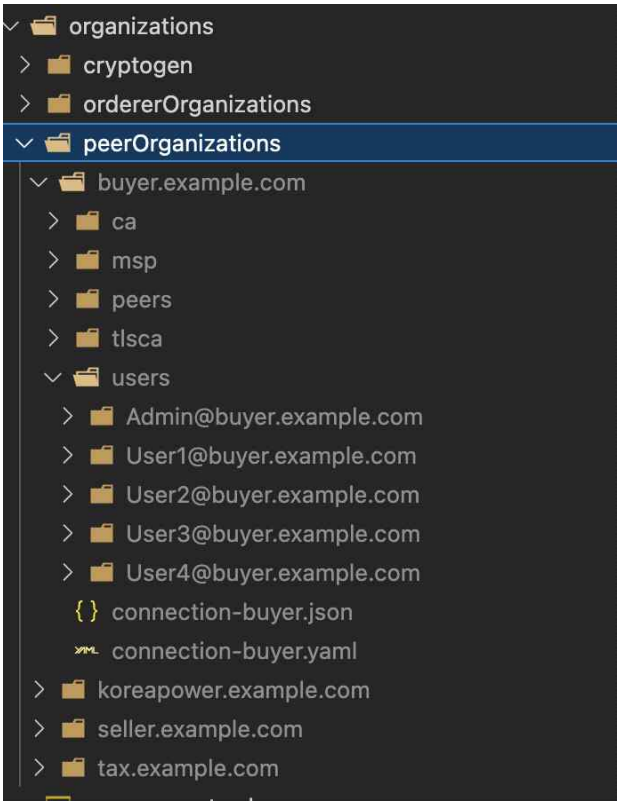


[그림 19] crpytogen.yaml

```
암호화 실행
/Users/kimcharless/github/AmeRes/AASing/AAS/../bin/cryptogen
Cryptogen을 이용해서 암호화 파일을 생성
사용자(판매자)의 암호화 파일 생성
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-seller.yaml --output=organizations
seller.example.com
+ res=0
사용자(구매자)의 암호화 파일 생성
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-buyer.yaml --output=organizations
buyer.example.com
+ res=0
국세청의 암호화 파일 생성
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-tax.yaml --output=organizations
tax.example.com
+ res=0
한국전력거래소의 암호화 파일 생성
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-koreapower.yaml --output=organizations
koreapower.example.com
+ res=0
오더러 암호화 파일 생성
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-orderer.yaml --output=organizations
+ res=0
Generating CCP files for Org1 and Org2
```

[그림 20] cryptogen 실행

[그림 20]는 Cryptogen을 사용해 config.yaml을 바탕으로 암호화 파일이 생성되는 모습이다.

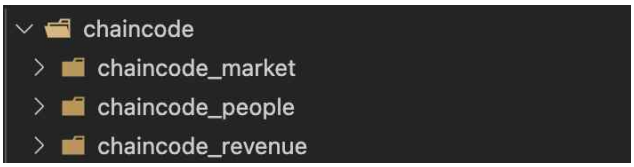


[그림 21] 암호화 파일

- 위를 사용해서 Organizations와 orderer의 암호화 파일을 생성할 수 있다.
  각 조직 사용자에 대한 msp가 생성이 되고 이 암호화 파일을 제출해서 블록체인 네트워크에 접근할 수 있다. 조직의 모든 사용자에 대한 암호화 파일이 형성되는 것을 확인할 수 있다.

4.2 체인코드

- 체인코드는 각 채널별로 하나씩 사용한다.



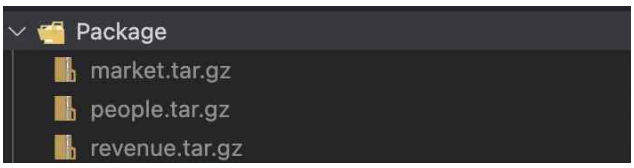
[그림 22] 체인코드

people	market	tax

type Asset struct {
DocType string `json:"docType"`
ID string `json:"ID"`
Seller string `json:"seller"`
Buyer string `json:"buyer"`
REC int `json:"REC"`
KRW int `json: "KRW"`
State string `json: "state"`
Time string `json: "Time"`
}
	type Asset struct {
	DocType string `json:"docType"`
	ID string `json:"ID"`
	REC int `json:"REC"`
	KRW int `json: "KRW"`
	Role string `json: "role"`
	}
		type Asset struct {
		DocType string `json:"docType"`
		ID string `json:"ID"`
		State string `json: "state"`
		Tax int `json: "tax"`
		}

CreateAsset() : asset 생성 UpdateAsset() : asset 업데이트 GetAssetHistory() : asset기록 열람 ReadAsset() : asset 열람		

체인코드의 구성은 다음과 같다.



[그림 23] 체인코드 패키지

체인코드는 tar.gz파일로 패키징되어 각 조직들에 설치된다.

4.3 블록체인 네트워크 실행

네 가지 작업이 완료되었으면 네트워크를 실행시킬 수 있다.

- 블록체인 네트워크 설정
- 도커 컴포즈 설정
- 암호화 파일 생성
- 체인코드 작성

네트워크 실행은 다음과 같은 작업들이 필요하다.

1. Docker-Compose 실행
2. 조직을 채널에 가입
3. 체인코드 패키징
4. 조직에 체인코드 설치
5. 설치된 체인코드에 대한 채널 구성원들의 동의

4.3.1 Docker-Compose 실행

[그림 24] 도커 컨테이너 실행

- 도커 컴포즈에 있는 도커 컨테이너들을 실행시킨다.
  도커 이미지가 올라오면서 컨테이너가 실행된다.

4.3.2 채널에 조직 가입

- 만들어둔 제네시스 블록에 대한 정보대로 조직을 채널에 가입시킬 수 있다.

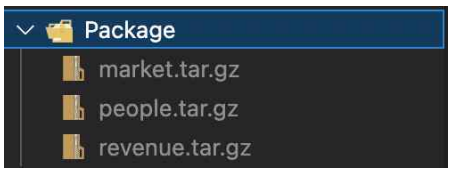
```
2022-09-26 02:15:11.907 KST 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-09-26 02:15:12.164 KST 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
buyer를 people에 가입중..
Using organization buyer
+ peer channel join -b ./channel-artifacts/people.block
+ res=0
2022-09-26 02:15:15.316 KST 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-09-26 02:15:15.573 KST 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
tax를 revenue에 가입중..
Using organization tax
+ peer channel join -b ./channel-artifacts/revenue.block
+ res=0
2022-09-26 02:15:18.678 KST 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-09-26 02:15:18.926 KST 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
koreapower를 market에 가입중..
Using organization koreapower
+ peer channel join -b ./channel-artifacts/market.block
+ res=0
2022-09-26 02:15:22.079 KST 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-09-26 02:15:22.378 KST 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
```

[그림 25] 채널 가입

조직을 채널에 가입시키고 조직의 앵커피어(전파 노드)를 설정해주면 체인코드를 설치할 준비가 완료된다.

4.3.3 체인코드 패키징, 설치

- 체인코드를 tar.gz파일로 압축한다.



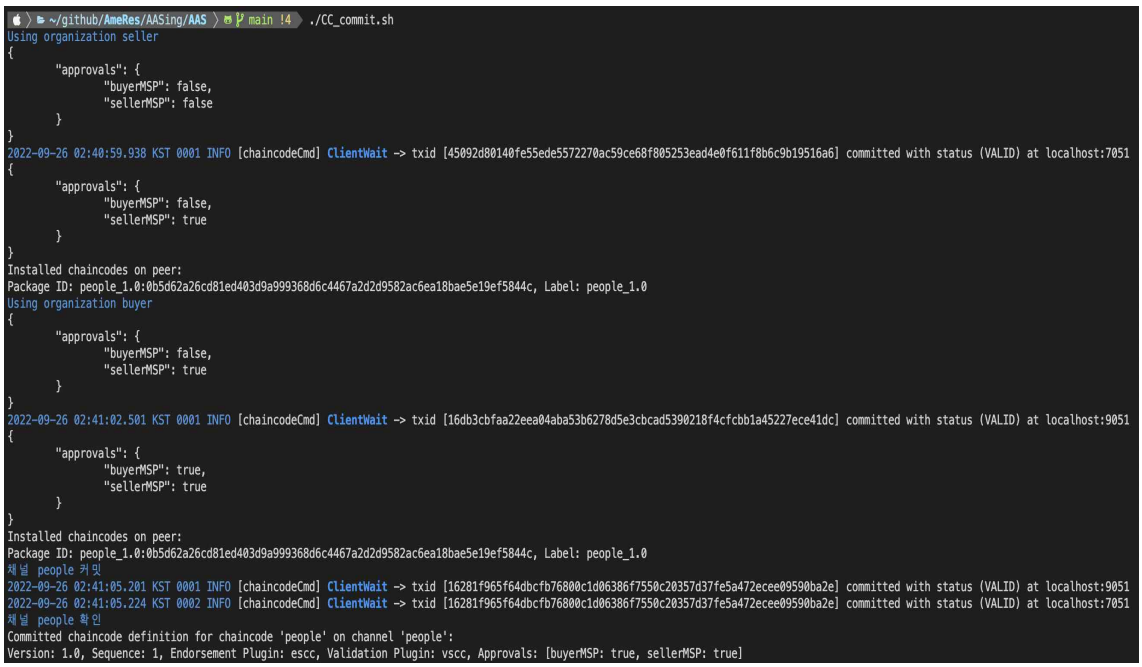
[그림 26] 패키징된 체인코드

체인코드를 패키징하여 조직에 설치한다.

```
Seller에 체인코드 설치
Using organization seller
2022-09-26 02:15:34.250 KST 0001 INFO [cli.lifecycle.chaincode] submitInstallProposal -> Installed remotely: response:<status:200 payload:"\nKpeople_1.0:0b5d62a26cd81ed403d9a999368d6c4467a2d2d9582ac6ea
2022-09-26 02:15:34.251 KST 0002 INFO [cli.lifecycle.chaincode] submitInstallProposal -> Chaincode code package identifier: people_1.0:0b5d62a26cd81ed403d9a999368d6c4467a2d2d9582ac6ea18bae5e19ef5844c
Buyer에 체인코드 설치
Using organization buyer
2022-09-26 02:15:34.407 KST 0001 INFO [cli.lifecycle.chaincode] submitInstallProposal -> Installed remotely: response:<status:200 payload:"\nKpeople_1.0:0b5d62a26cd81ed403d9a999368d6c4467a2d2d9582ac6ea
2022-09-26 02:15:34.409 KST 0002 INFO [cli.lifecycle.chaincode] submitInstallProposal -> Chaincode code package identifier: people_1.0:0b5d62a26cd81ed403d9a999368d6c4467a2d2d9582ac6ea18bae5e19ef5844c
tax에 체인코드 설치
Using organization tax
2022-09-26 02:15:34.580 KST 0001 INFO [cli.lifecycle.chaincode] submitInstallProposal -> Installed remotely: response:<status:200 payload:"\nLrevenue_1.0:e50a168643ae87f579305e89145e0d0f2bd44ff5e2e1e34
2022-09-26 02:15:34.580 KST 0002 INFO [cli.lifecycle.chaincode] submitInstallProposal -> Chaincode code package identifier: revenue_1.0:e50a168643ae87f579305e89145e0d0f2bd44ff5e2e1e34585e5a47ef8a8fe52
koreapower에 체인코드 설치
Using organization koreapower
2022-09-26 02:15:34.738 KST 0001 INFO [cli.lifecycle.chaincode] submitInstallProposal -> Installed remotely: response:<status:200 payload:"\nKmarket_1.0:4af97bf2b3b2df6ef090c0b81e397dd8b1f837b98b660485
2022-09-26 02:15:34.738 KST 0002 INFO [cli.lifecycle.chaincode] submitInstallProposal -> Chaincode code package identifier: market_1.0:4af97bf2b3b2df6ef090c0b81e397dd8b1f837b98b66048513781 6f229c4eef4
```

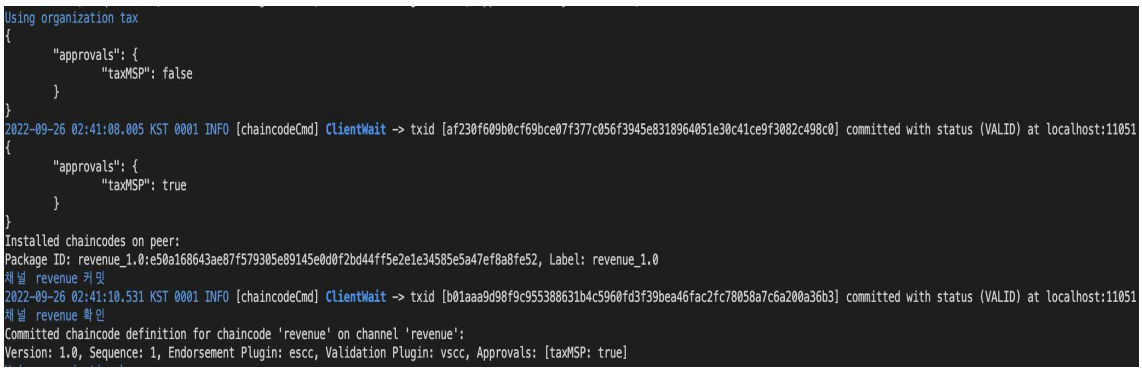
[그림 27] 체인코드 설치

- 체인코드 설치가 완료되면 채널 구성원들의 동의가 필요하다.
  채널구성원들이 동의를 하면 체인코드 실행준비가 완료된다.



[그림 28] people 체인코드 커밋

- 체인코드는 각 채널 구성원의 2/3이상이 동의해야 성공적으로 실행이 가능하다.
  위 그림은 채널 people의 체인코드 people.tar.gz에 동의하는 과정을 보여준다.
  seller와 buyer가 체인코드의 동의하고 성공하면 people 체인코드가 성공적으로 커밋되었음을 확인할 수 있다.



[그림 29] revenue체인코드 커밋

- 조직의 구성원이 한 명일 경우에는 한 명의 동의로 체인코드 설치가 가능하다.

블록체인 네트워크의 실행, 체인코드 설치 및 커밋이 완료되었으므로 블록체인 네트워크에 접근할 수 있다.

4.4 Back-End

- 본 과제에서는 블록체인 네트워크에 접근할 수 있는 Spring Boot를 활용한 Rest-Api를 구축하여 사용하였다.

- Java로 만들어진 Gateway를 활용하여 블록체인 네트워크에 접근할 수 있다. 네트워크에 접근하기 위해서는 gateway라이브러리가 필요하다.

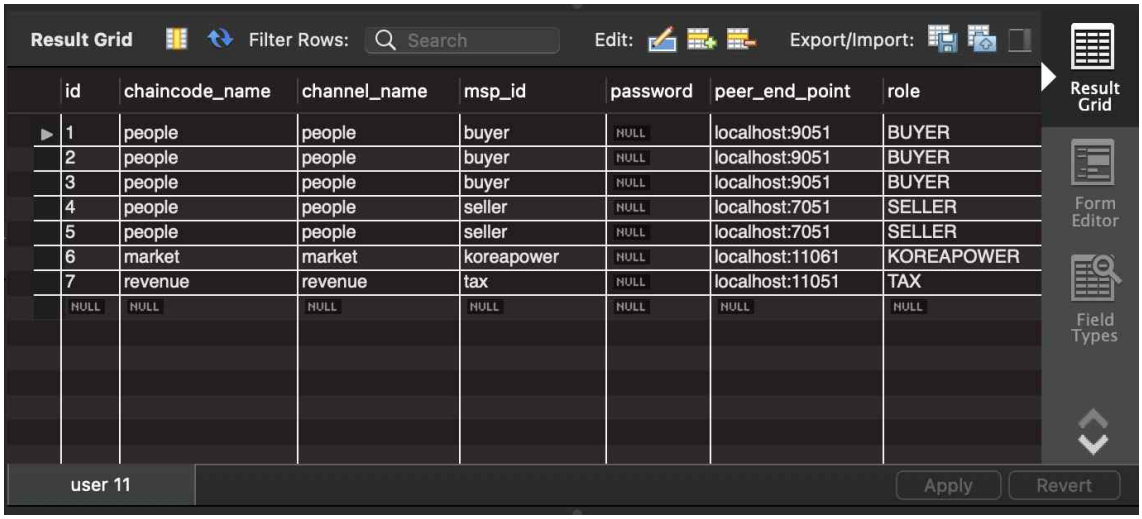
- 다음 라이브러리를 추가해준다.

```
implementation 'org.hyperledger.fabric:fabric-gateway:1.1.0'
implementation 'io.grpc:grpc-netty-shaded:1.49.0'
```

- 블록체인 네트워크에 접근하기 위해서는 조직 사용자들의 정보가 필요하다.

필요 파일	내용
MspID	msp(membership service provider)의 id
cert	사용자의 cert.pem 키
tls	사용자의 ca.crt 키
key	사용자의 priv_sk키
peerEndPoint	사용하는 조직의 peerEndPoint 포트
채널 이름	속한 채널 이름
체인코드 이름	사용할 체인코드 이름

Front-page에서 로그인에 성공하면, 위 정보들의 경로를 Mysql 데이터 서버에서 읽어서 가져와 준다.



id	chaincode_name	channel_name	msp_id	password	peer_end_point	role
1	people	people	buyer	NULL	localhost:9051	BUYER
2	people	people	buyer	NULL	localhost:9051	BUYER
3	people	people	buyer	NULL	localhost:9051	BUYER
4	people	people	seller	NULL	localhost:7051	SELLER
5	people	people	seller	NULL	localhost:7051	SELLER
6	market	market	koreapower	NULL	localhost:11061	KOREAPOWER
7	revenue	revenue	tax	NULL	localhost:11051	TAX
NULL	NULL	NULL	NULL	NULL	NULL	NULL

[그림 31] 데이터베이스

- 데이터베이스에는 블록체인 네트워크 통신에 필요한 파일들을 가지고 있다.

- 통신에 대한 정보가 준비되었으면 블록체인 네트워크와 통신하는 Contract 객체가 필요하다.

```
    public EasilyConnect(MemberDto memberDto) throws Exception {
        this.memberDto = memberDto;
        var channel = newGrpcConnection();
        var builder =
Gateway.newInstance().identity(newIdentity()).signer(newSigner()).connection(channel)
                .evaluateOptions(options -> options.withDeadlineAfter(5, TimeUnit.SECONDS))
                .endorseOptions(options -> options.withDeadlineAfter(15, TimeUnit.SECONDS))
                .submitOptions(options -> options.withDeadlineAfter(5, TimeUnit.SECONDS))
                .commitStatusOptions(options -> options.withDeadlineAfter(1, TimeUnit.MINUTES));
        var gateway = builder.connect();
        var network = gateway.getNetwork(this.memberDto.getChannelName());
        this.contract = network.getContract(memberDto.getChaincodeName());
    }
```

- Grpc연결을 만든 후에 필요한 정보들을 함수에 넣어서 새로운 Contract를 만든다 .

```
var result = contract.evaluateTransaction("ReadAsset",id);

contract.submitTransaction("CreateAsset",....);
```

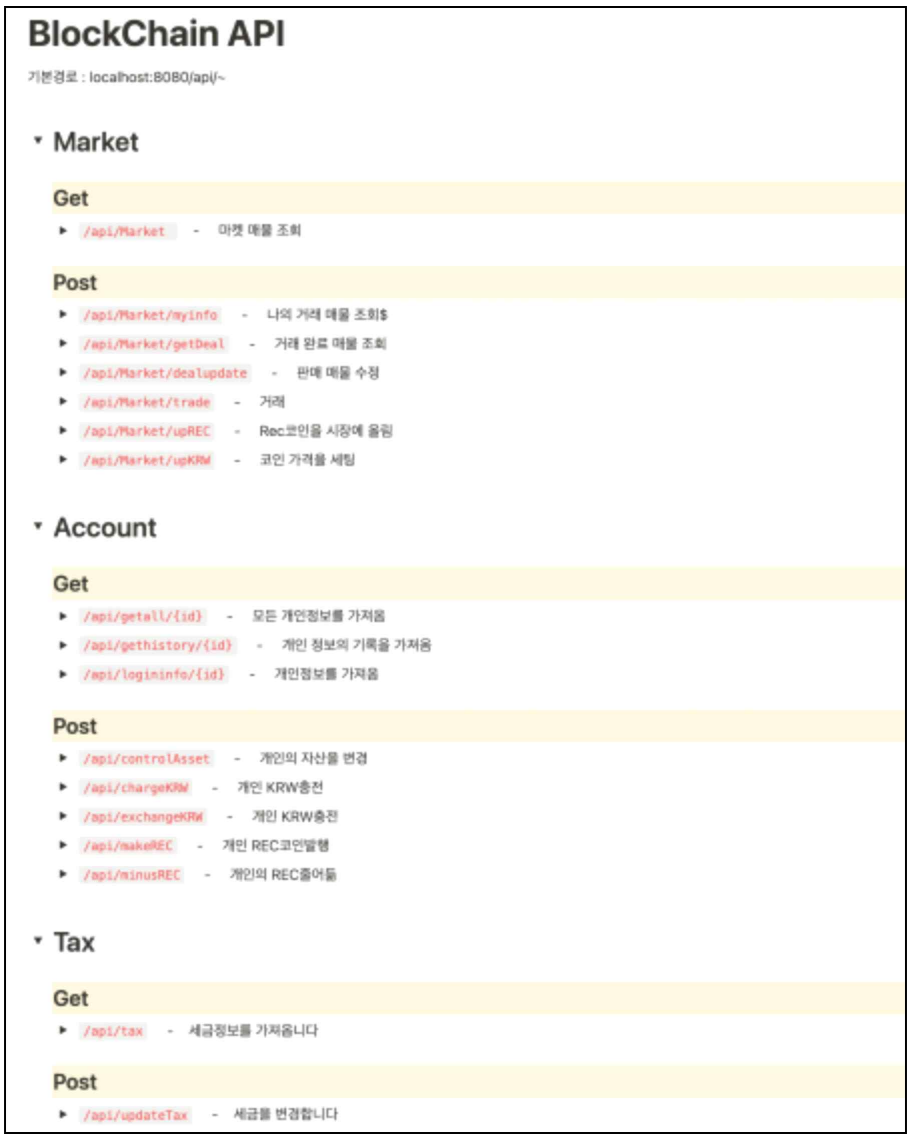
- 해당 contract를 evaluateTransction나 submitTransaction함수를 사용하여 블록체인 네트워크의 체인코드를 실행할 수 있다.

- 서버의 구조는 다음과 같다.

Controller	**AccountApiController**	계정 정보에 대한 API /api/getAll /api/getHistory /api/logininfo /api/makeREC /api/exchangeKRW /api/minusREC /api/ChargeKRW /api/controlAsset
	MarketApiController	**마켓 정보에 대한 API** /api/Market /api/Market/myinfo /api/Market/getDeal /api/Market/dealupdate /api/Market/upKRW /api/Market/trade
	TaxApiController	**세금 정보에 대한 API** /api/tax /api/updateTax
Dto	**AssetDto**	계정 Dto
	MarketDto	매켓 Dto
	HistoryDto	기록 Dto
	TaxDto	세금 Dto
	TradeDto	거래내용 Dto
	MemberDto	통신 계정 정보 Dto
Connect	**EasilyConnect**	**블록체인 contract 생성 함수** EasilyConnect newGrpcConnection() newIdentity() newSigner() getPrivateKeyPath() **블록체인 네트워크 통신 함수** GetAllAsset() GetTax() GetAssetById() getAssetHistory CreateAsset() UpdateAsset() GetMarket()

- API는 Notion과 PostMan을 사용해 정리하였다

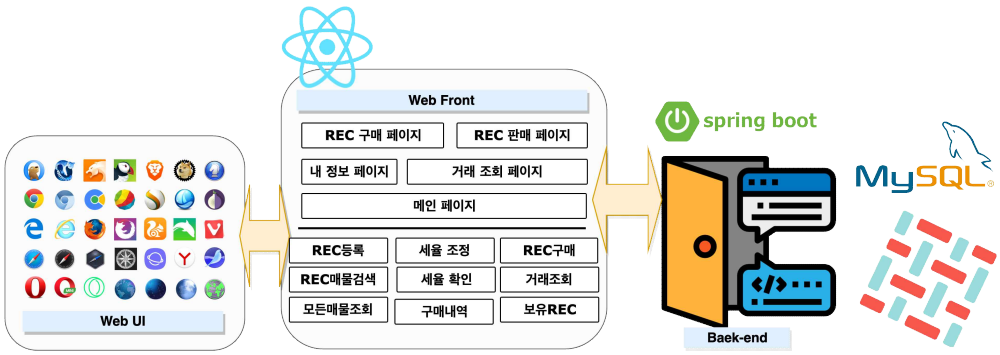
- **https://www.notion.so/BlockChain-API-6c94cc49a9b2482d940bc07a6102ce23**



[그림 34] 블록체인 api

4.5 Front-End

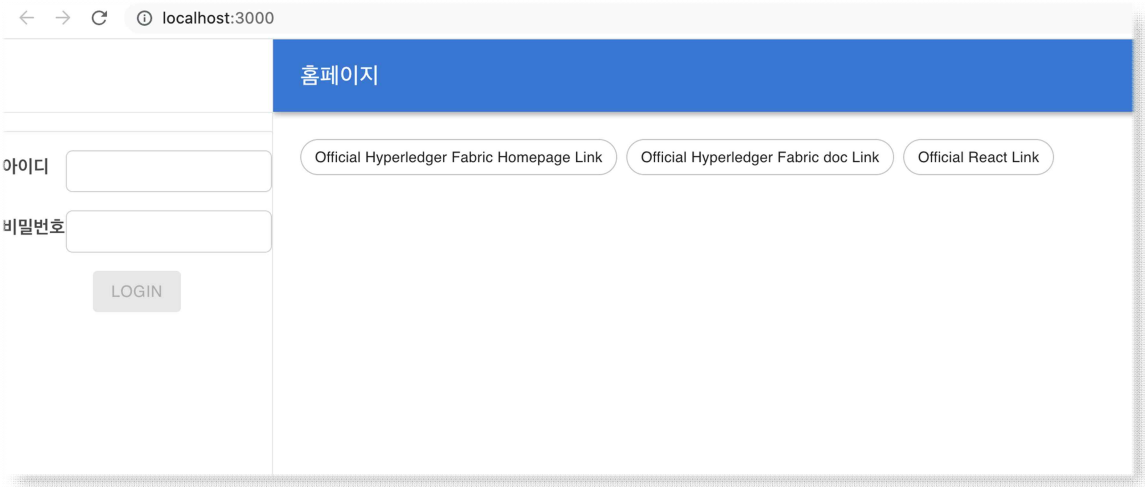
4.5.1 구조도



[그림 35] 구조도

- 본 팀은 front-end를 Reactjs를 사용해서 작성했다. 그리고 react ui 라이브러리인 material ui를 활용하여 front-end를 만들었다. Reactjs를 사용함으로써 컴포넌트 기반으로 front-end를 구성하였다.

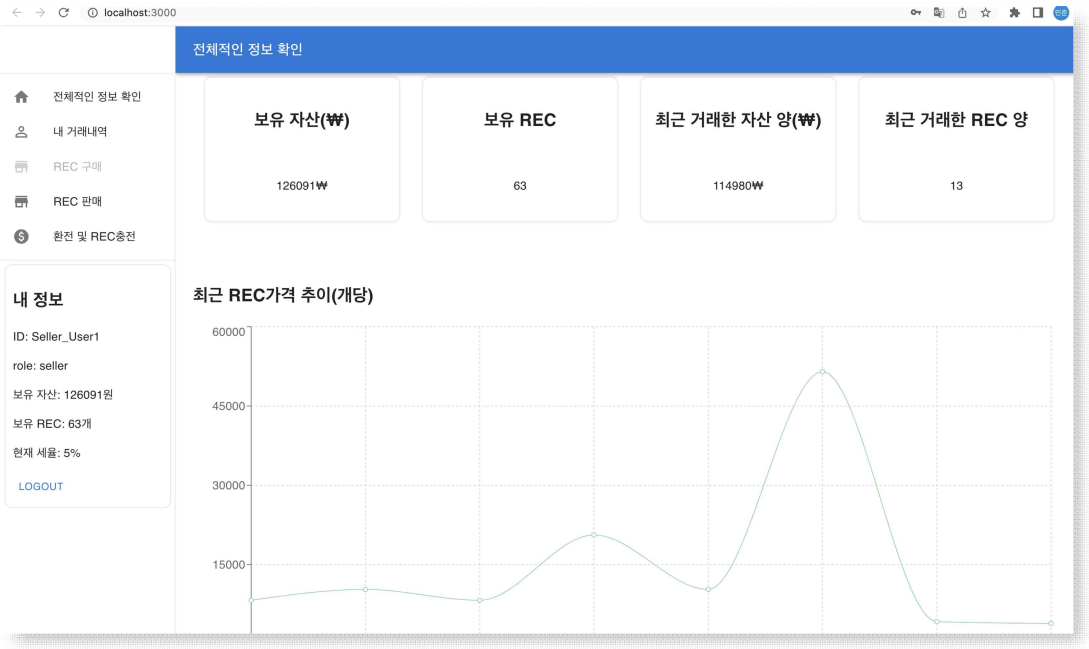
4.5.2 Buyer



[그림 36] 최초 접속 화면

- 최초로 페이지에 접근했을 때의 화면이다. 좌측에는 로그인을 할 수 있는 컴포넌트가 위치하고 있고 상단에는 현재 페이지의 이름이 나타나 있다. 중앙 메인 컴포넌트에는 hyperledger fabric의 공식홈페이지, doc와 react의 공식홈페이지로 이동할 수 있는 링크가 위치하고 있다. 일단 비밀번호는 더미데이터로 아무 값이나 입력해도 로그인이 가능하도록 하였고, 로그인은 Seller_User1, Seller_User1, Buyer_User1, Buyer_User2, koreapower_admin, tax_admin 이렇게 여섯 가지로 로그인이 가능하도록 설계하였다. 이 중 앞 네 가지는 사용자고 뒤 두 가지는

관리자이다.



[그림 37] Seller_User 로그인 화면

ID	메인 페이지	내 거래내역	REC 판매	환전 및 REC 충전
seller	사용자의 정보를 개략적으로 보여주고 REC 개당 가격의 추이를 차트로 알려준다.	사용자의 거래내역을 시간 순으로 보여준다.	보유 중인 REC를 시장에 등록하고 가격을 등록한다.	REC를 충전하고 자산을 환전 할 수 있다.

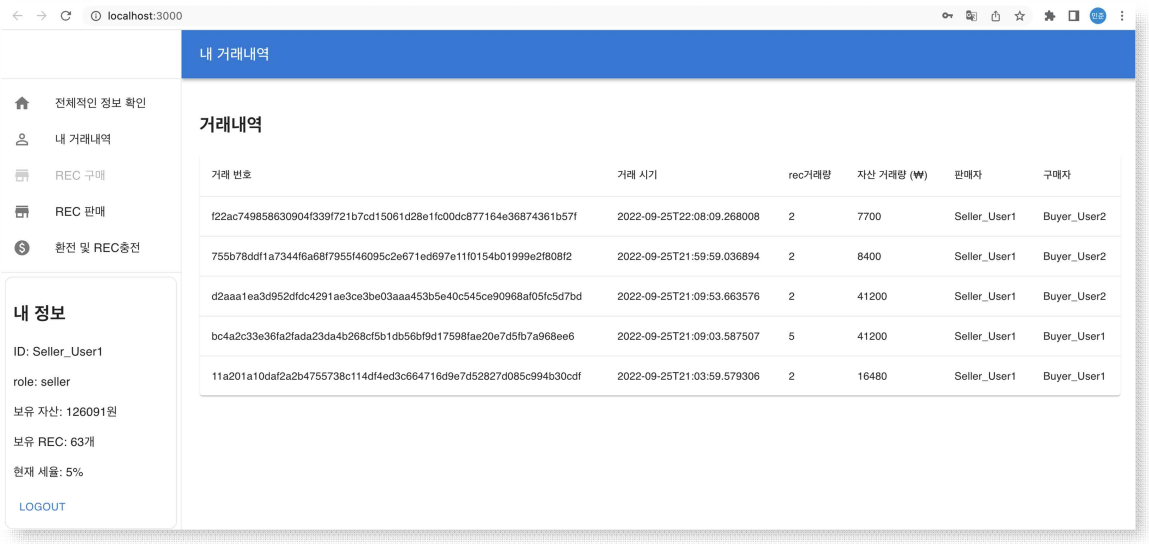
- 로그인 이후 위와 같이 로그인 컴포넌트는 사라지고 사용자의 정보를 표시하는 컴포넌트가 생기는 것을 볼 수 있다

  로그인 이후에는 사용자에 따라서 좌측에 메뉴 목록이 생긴다. rec 판매자의 경우에는 전체적인 사용자의 정보를 볼 수 있는 "전체적인 정보 확인" 메뉴, 내 거래내역을 알 수 있는 "내 거래내역" 메뉴, 판매할 rec를 등록할 수 있는 "rec 판매" 메뉴, rec를 충전하고 자산을 환전할 수 있는 "환전 및 REC충전"을 볼 수 있다. (구매는 불가능하므로 rec 구매 버튼은 비활성화 된다.)

  메인 페이지를 보면 사용자의 보유 rec, 자산, 최근에 거래한 총 rec, 자산을 볼 수 있다. 그 아래에는 차트를 통해서 최근 시장에서 거래된 rec 개당 가격의 추이를 확인할 수 있다. 본 차트는 recharts 라이브러리를 활용하였다.

- 유저가 최근 거래한 자산의 총 양과 최근 거래한 rec의 총 양을 알아내기 위해 POST /api/Market/myinfo을 요청하여 데이터를 받아온다.

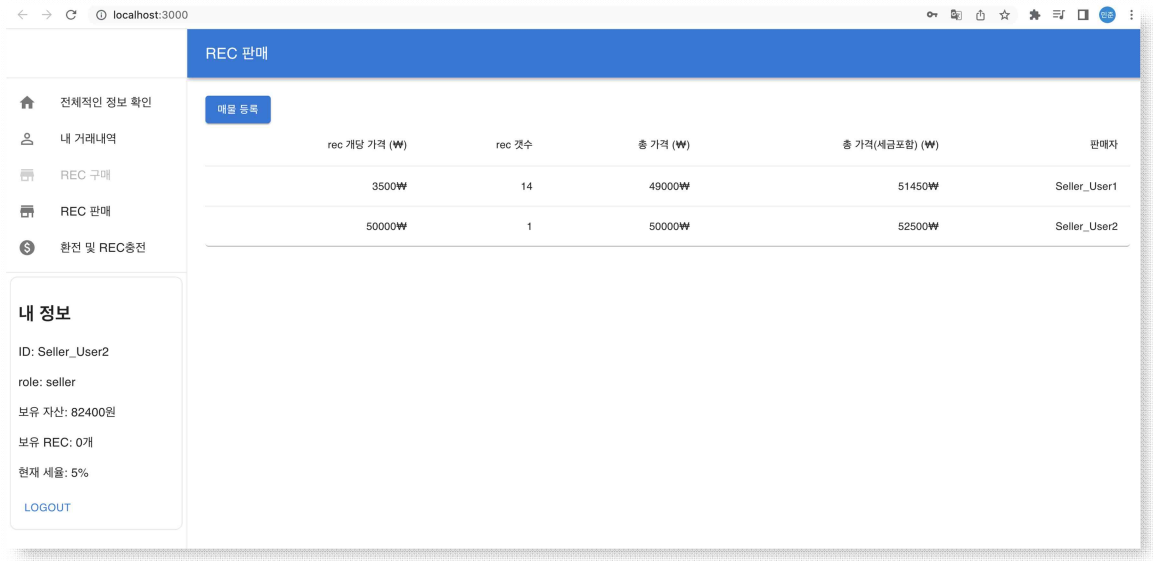
- 유저가 최근 거래한 자산의 총 양과 최근 거래한 rec의 총 양을 알아내기 위해 POST /api/Market/myinfo을 요청하여 데이터를 받아온다.



[그림 38] 내 거래내역 페이지

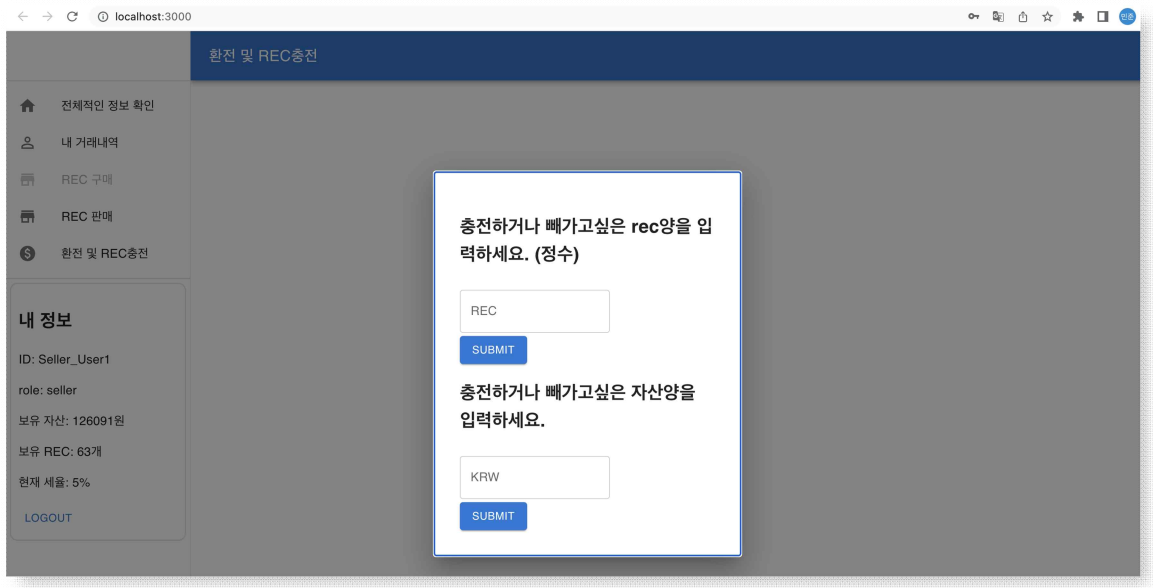
- 내 거래내역을 누르면 [그림 37]과 같은 페이지가 나타나게 된다. 중앙 페이지에 표가 나타나게 되는데, 표에서 사용자가 거래했던 거래 별로 거래 번호, 거래 시기, rec거래량, 자산 거래량, 판매자, 구매자를 알 수 있다. 거래 번호는 거래 체결시 sha256해시값이 생성된다.

- 거래내역 데이터를 받아오기 위하여 POST /api/Market/myinfo 요청을 하여 데이터를 받아온다.



[그림 39] 매물 등록 페이지

- 현재 시장에 등록된 매물들이 보여준다. 판매자는 위 매물들의 가격을 참고하여 rec의 개당 가격을 설정하거나 자신이 소유하고 있는 rec를 매물로 등록할 수 있다.
- GET /api/Market을 통하여 시장에 등록된 매물 데이터를 가져올 수 있다.



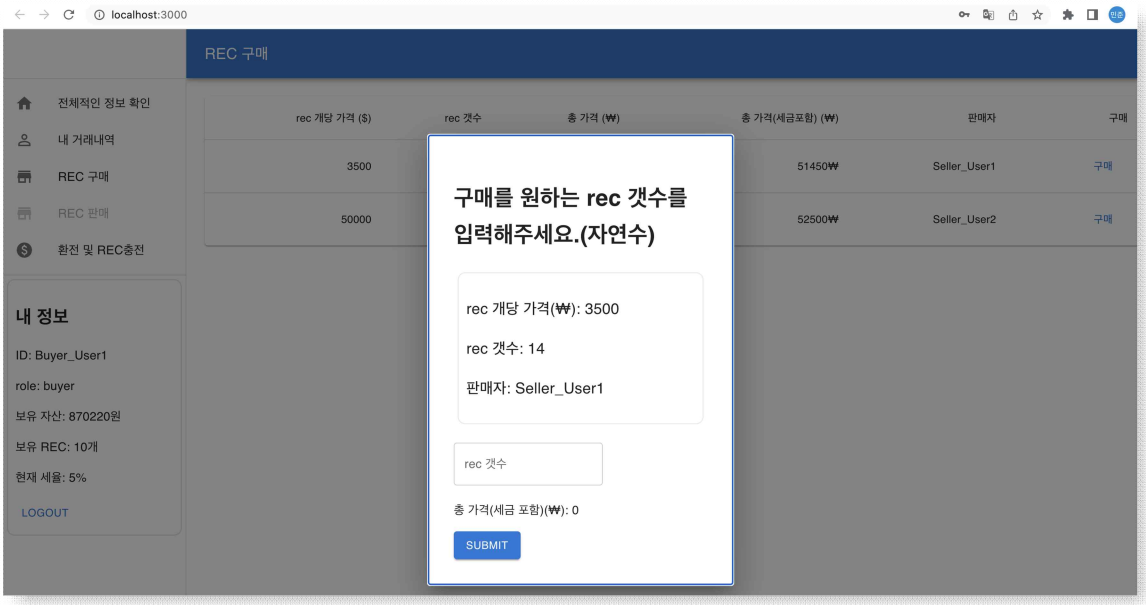
[그림 40] 환전 및 REC충전 메뉴

- 그림 5와 같이 Modal을 통해서 REC를 충전하고 자산도 충전할 수 있다. 등록 이후에는 내 정보 컴포넌트에서 rec와 자산이 바뀐 것을 바로 확인해 볼 수 있

다.

- POST /api/makeREC을 통하여 입력한 REC 양이나 자산 양을 백엔드 서버에 전송한다.

4.5.3 Buyer



[그림 41] REC 구매 메뉴

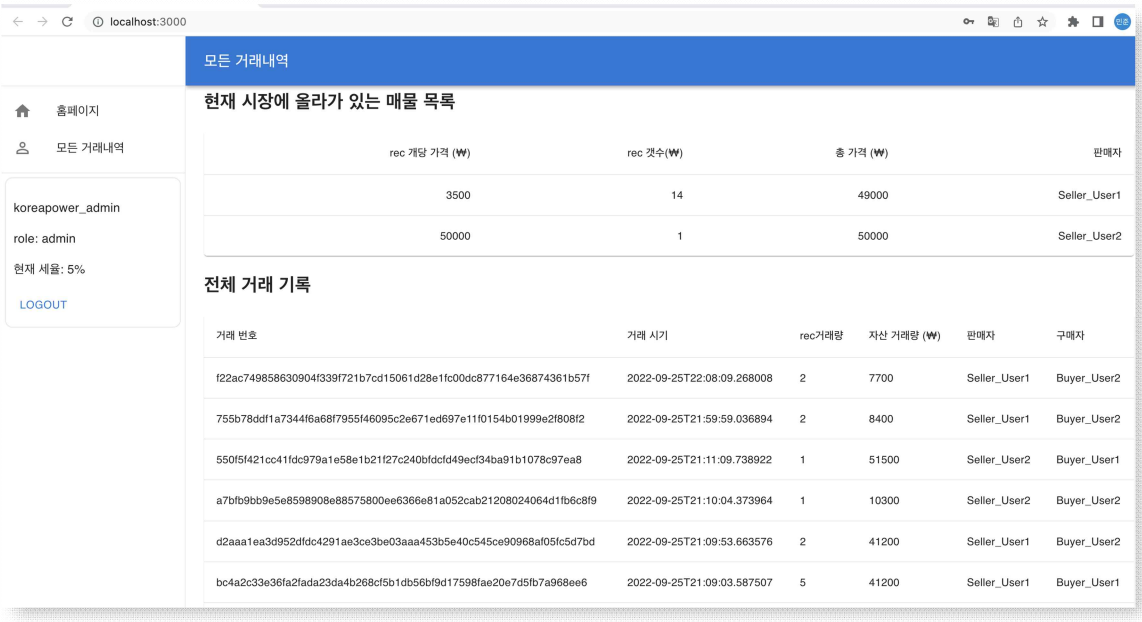
ID	메인 페이지	내 거래내역	REC 구매	환전 및 REC 충전
buyer	사용자의 정보를 개략적으로 보여주고 REC 개당 가격의 추이를 차트로 알려준다.	사용자의 거래내역을 시간 순으로 보여준다.	시장에 등록된 매물을 확인하고 REC를 구매한다.	REC를 충전하고 자산을 환전 할 수 있다.

- 구매자로 로그인한 후에 REC 구매 메뉴를 통해서 REC를 구매할 수 있다. 위 Modal에 원하는 개수를 입력해서 REC를 구매할 수 있다. 자신의 자산보다 많이 구매하려고 하면 Modal창을 통하여 입력을 제한한다.

- POST /api/Market/trade를 통해서 백엔드 서버에 데이터를 전송하여 구매자의 데이터에 REC의 개수를 업데이트하고, 시장의 REC 개수도 업데이트한다.

- 그 외 기능은 Seller와 동일하다.

4.5.4 koreapower_admin(전력거래소)



[그림 42] 일반 관리자의 모든 거래내역 메뉴

- 전력거래소로 로그인한 후에는 모든 거래 내역 메뉴에서 시장에 올라온 매물과 모든 사용자의 거래내역을 볼 수 있다.

- 이때, POST /api/Market/getDeal을 통하여 현재 시장에 올라와 있는 매물 데이터를 가져오고, GET /api/Market 명령어를 시장에 저장되어있는 모든 사용자의 거래 내역을 가져온다.

rec 개당 가격 (₩)	rec 갯수(₩)	총 가격 (₩)	판매자
10000	2	20000	Seller_User1
10000	5	50000	Seller_User2

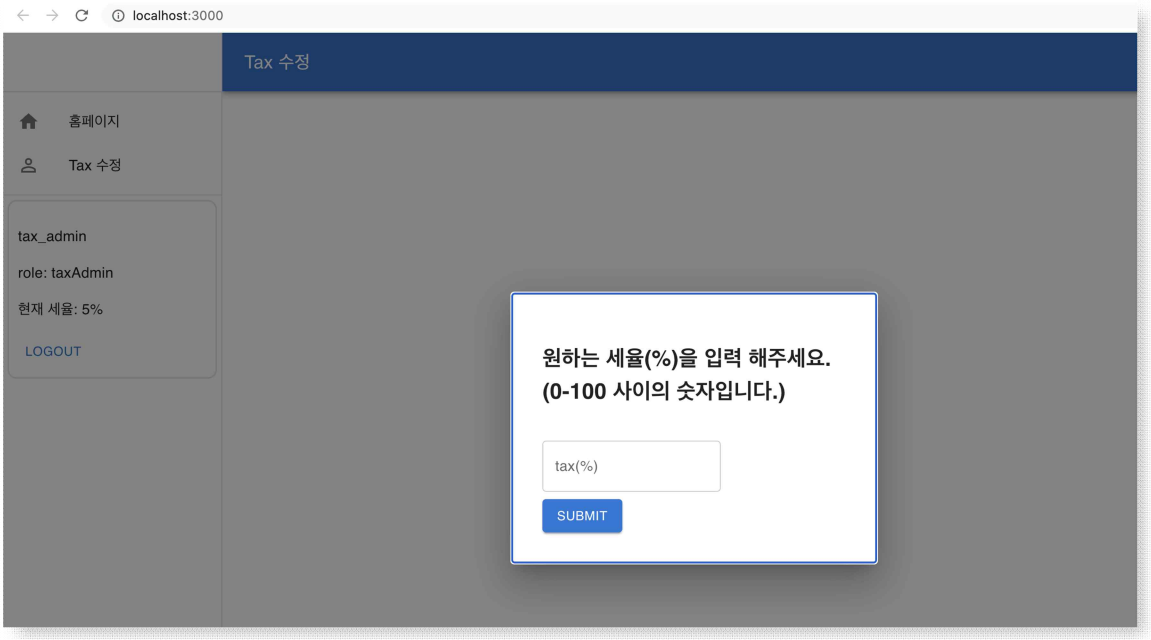
- 현재 판매 중인 매물의 내용을 확인할 수 있다.

전체 거래 기록

거래 번호	거래 시기	rec거래량	자산 거래량 (₩)	판매자	구매자
a83b5e2fe56b676f2652c52494a837f5e52c63a6e6d4adf1abdb52a4968f5fb3	2022-09-26T04:36:24.821796	3	30900	Seller_User1	Buyer_User2

- 체결된 거래기록을 확인할 수 있다. 생성된 해시값과 거래 시간, 그리고 거래정보를 확인할 수 있다.

4.5.5 Tax(국세청)



[그림 45] tax관리자의 Tax수정 메뉴

- 국세청 관리자로 로그인하면 tax를 정할 수 있다. 이때 tax는 % 단위로 적용된다. 0~100 사이의 숫자를 입력하지 않으면 Modal창이 나타나 입력을 제한하게 된다.

- POST /api/updateTax을 통해서 update 하고 싶은 세율을 보내고 백엔드 서버에서 tax를 업데이트하게 된다

5. 결론 및 향후 연구 방향

5.1 결론

- 본 팀은 프라이빗 블록체인인 하이퍼래저 패브릭(Hyperledger-fabric)을 사용하여 블록체인 네트워크를 만들고 spring boot를 활용해 Rest-api를 구축하여 안전한 Rec거래 플랫폼을 제작하였다.

- 현재 REC거래시장에서 사용되는 분산된 데이터베이스에 비해 블록체인 네트워크 사용으로 신뢰성 확보가 가능했다.

- 프라이빗 블록체인을 사용해 chain-code를 활용한 smart-contract가 가능해지며 사용자의 신뢰성 있는 권한 조정을 할 수 있었다.

- 판매자, 구매자, 전력거래소의 모든 기록에 블록 id가 붙으며 위변조할 수 없는 거래를 생성할 수 있었다.

5.2 향후 연구 방향

- 본 팀의 블록체인 네트워크는 구매자, 판매자, 거래시장이 모두 다른 채널 다른 체인코드에 분산되어있다. 기존 데이터베이스에 비해서는 훨씬 안전하지만 기존의 데이터베이스 분산 문제를 완전하게 해결하지는 못하였다. 같은 체인코드에 서로 다른 형식의 데이터를 보관하며 구매자, 판매자, 거래시장을 같은 채널에 넣고 같은 체인코드를 사용하여 거래할 수 있도록 해야 함을 과제 진행하면서 알게 되어 아쉬움이 있었다.

- 실제 거래 플랫폼에서는 구매자나 사용자의 로컬PC에 가지고 있는 인증서를 통하여 서버에 거래를 제출해야 하지만, 본 팀의 서버에서는 데이터베이스에서 유저의 정보를 읽어 와서 하는 방식이다. 좀 더 현실적인 거래를 위해서는 로컬PC에서 직접 인증서의 키를 보내는 방식을 취할 필요가 있다.

- 거래량이 계속해서 누적되면 한 개의 오더러만으로는 트랜잭션을 감당할 수 없어 분산이 필요하다. 조직의 사용자가 누적됨에 따라 오더러의 수를 늘려줄 필요가 있다.

- 전력공급자가 전력을 생산하는 곳에서 직접 전력을 감지하고 자동으로 REC

코인을 생성해주는 인프라 구축를 확충하는 것이 REC의 신뢰성을 높이는데 필요해 보인다.

6. 참고 문헌

[1] Hyperledger-foundation [Online]. Available: https://hyperledger-fabric.readthedocs.io/ko/latest

[2] Reactjs [Online]. Available: https://reactjs.org

[3] spring-boot [Online]. Available : https://spring.io/projects/spring-boot

[4] hyperledger gateway [Online]. Abailable : https://github.com/hyperledger/fabric-gateway-java

[5] hyperledger gateway [Online]. Abailable : https://github.com/hyperledger/fabric-sdk-go